Automatic and Harmless Regularization with Constrained and Lexicographic Optimization: A Dynamic Barrier Approach

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many machine learning tasks have to make a trade-off between two loss functions, typically the main data-fitness loss and a regularization or auxiliary loss. The most widely used approach is to optimize the linear combination of the objectives, which, however, requires manual tuning of the combination coefficient and is theoretically unsuitable for non-convex functions. In this work, we consider constrained optimization as a more principled approach for trade-off two losses, with a special emphasize on *lexicographic optimization*, a degenerated limit of constrained optimization which optimizes a secondary loss inside the optimal set of a main loss. We propose a *dynamic barrier gradient descent* algorithm which provides a unified solution of both constrained and lexicographic optimization. We establish the convergence of the method for general non-convex functions. Through comprehensive experiments on real-world deep learning tasks, we show that 1) lexico optimization provides a parameter-free approach to incorporating side loss function without hurting the main objective, and 2) constrained combined with leixco optimization provides an automatic approach to profiling Pareto sets, especially non-convex Pareto sets on which linear combination methods fail.

1 Introduction

Although machine learning (ML) has been typically conceptualized as optimizing a single objective function, most practical ML tasks actually involve trading off two or more objective functions, such as the data fitness function vs. a regularization or auxiliary loss. Let f and g be two objectives functions of interest on $\mathbb{R}^d$. A principled way to trade-off f and g is through constrained optimization:

$$\min_{\theta \in \mathbb{R}^d} f(\theta) \quad s.t. \quad g(\theta) \leq c, \tag{1}$$

where c is a threshold. When varying c, the solution of (1) fills the Pareto frontier of (f, g), providing different trade-offs on f, g. In addition, (1) naturally arises when g is an important constraint that should be controlled explicitly, e.g., in terms of safety, fairness, and other trustworthy measures.

However, compared to unconstrained optimization, constrained optimization (1) has been much less widely used in practical machine learning. In fact, perhaps the most common approach to handling (1) is to transform it into the unconstrained optimization of the *linear combination* of f and g,

$$\min_{\theta \in \mathbb{R}^d} f(\theta) + \lambda g(\theta), \tag{2}$$

where the trade-off between f and g is controlled by a weight coefficient λ, instead of the threshold parameter c. A folklore argument is that λ can be viewed as the Lagrange multiplier of (1), and hence (1) can be mapped into (2) with properly selected λ.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Unfortunately, despite being broadly used, (2) is insufficient to fully replace (1) and suffers from a number of disadvantages in both theory and practice:

• *Interpretability.* λ provides a less intuitive parameter to select than c, since it depends on the relative scale of f and g, whose range is problem-dependent and needs to be optimized as a hyper-parameter. In comparison, the threshold c can be specified as a tolerance parameter when g is a metric that users want to control explicitly.

• *Pareto Coverage.* When f, g are non-convex functions, (1) provides a strictly broader class of problems (2), because for some c, there may exist no $\lambda \in \mathbb{R}$, such that (1) and (2) are equivalent. From the multi-objective optimization (MOO) perspective, this is related to the fact that (2) can only capture the convex envelop of the Pareto set while varying λ, while (1) provides all the strongly Pareto optimal points by varying c.

• *Invariance.* The constrained optimization (1) is invariant to arbitrary monotonically increasing maps, say ψ and ψ', on f and g, that is, we obtain the equivalent problem if we replace f with $\psi \circ f$, and g with $\psi' \circ g$, and c with $\psi'(c)$. In comparison, applying nonlinear transforms on f and g in (2) yields fundamentally different problems.

• *Harmless Regularization.* In many practical cases, one of the objectives (say f) is of secondary importance w.r.t. the other (g), in sense that we are interested in minimizing f only when g is fully optimized; this can be formulated as the following *lexicographic optimization* problem:

$$\min_{\theta \in \mathbb{R}^d} f(x) \quad s.t. \quad g(\theta) \leq g^* \qquad \text{where} \qquad g^* := \inf_{\theta \in \mathbb{R}^d} g(\theta), \tag{3}$$

where we minimize f inside the optimum set $\{\theta \colon g(\theta) \leq g^*\}$ of g. This can be viewed as a special case of (1) with the minimum threshold $c = g^*$ (which is unknown before hand). But can not be captured by (2) with any fixed λ. Compared with constrained optimization (1), the lexico problem eliminates the need for setting the threshold c, and hence provides a handy, parameter-free approach for incorporating auxiliary losses without hurting the main loss. As our empirical results show, this finds useful in numerous deep learning tasks, e.g., fairness ML and semi-supervised learning.

Our Contribution This work conducts a joint study on the constrained and lexico optimization problems. We propose a general local descent algorithm which offers a unified solution to both problems. By placing a *dynamic barrier* constraint on the search direction at iteration, the algorithm finds a trajectory towards the optimal solution by properly balancing the minimization of f and the constraint on g. We investigate the theoretical properties of our method for general non-convex functions, and establish convergence guarantees for both constrained and lexico optimization (Section 3.1-3.2). In comparison, we show that the methods based on optimizing the linear combination (2) is fundamentally unsound as an approach to (1) for non-convex functions (Section 3.3). Empirically, we show that our method provides an efficient approach to approximating Pareto sets and incorporating side information in a variety of deep learning tasks.

2 Main Method

We propose a simple gradient-based algorithm for solving both the constrained optimization (1) and lexicographic optimization (3) in a unified way. The method performs iterative updates of form

$$x_{t+1} \leftarrow x_t - \epsilon_t v_t,$$

where ϵ_t is a step size and v_t is an update direction to be chosen to balance the loss minimization on f and constrained satisfaction on g. The choice of v_t should satisfy the following desiderata:

1) When the constraint is not satisfied ($g(\theta) > c$), we should mainly focus on decreasing g to meet the constraint as fast as possible; meanwhile, f should act as a secondary objective in this phase, which means that it should be minimized only given that it does not conflict with the descent of g.

2) When the constraint is met ($g(\theta) \leq c$), we should prioritize to minimize f, which is made possible in general only if we allow g to increase. However, the increasing rate of g should be properly controlled, so that θ stays inside or nearby the feasible set while we minimize f.

It turns out that both properties can be achieved by selecting v_t by the following optimization:

$$v_t = \arg\min_{v \in \mathbb{R}^d} \left\{ \|\nabla f(\theta_t) - v\|^2 \;\; s.t. \;\; \nabla g(\theta_t)^\top v \geq \phi(\theta_t) \right\}, \tag{4}$$

where we want v_t to be as close to $\nabla f(x_t)$ as much as possible (and hence decrease f), but subject to a lower bound on the inner product $\nabla g(\theta_t)^\top v_t$ that ensures that the change of g is properly controlled depending on the location of θ; here the lower bound $\phi\colon \mathbb{R}^d \to \mathbb{R}$ is a *dynamic barrier* function which trade-offs loss minimization with constraint satisfaction by controlling the inner product between $\nabla g(\theta_t)$ and v_t. To achieve the desiderata on v_t, we should $\phi(\theta_t)$ to have the same sign as $g(\theta_t) - c$, so that the constraint $\{\theta\colon\ g(\theta) \le c\}$ is equivalent to $\{\theta\colon\ \phi(\theta) \le 0\}$, that is,

$$\operatorname{sign}(\phi(\theta)) = \operatorname{sign}(g(\theta) - c), \tag{5}$$

where $\operatorname{sign}(x) = x/\left|x\right|$ for $x \neq 0$ and $\operatorname{sign}(0) = 0$, In this way, when $g(\theta_t) > c$ (outside of feasible set), the constraint is $\nabla g(\theta_t)^\top v_t \ge \phi(\theta_t) > 0$, which ensures that g decreases strictly outside of the feasible set; when $g(\theta_t) = c$ (on the boundary), the constrain reduces to $\nabla g(\theta_t)^\top v_t \ge \phi(\theta_t) = 0$, meaning that we want to decrease f subject to that g does not increase; when $g(\theta_t) < c$ (interior of the feasible set), the lower bound on $\nabla g(\theta_t)^\top v_t$ is negative, allowing g to increase; in this case, the optimal solution in (4) can be shown to have a positive inner product with $\nabla f(\theta_t)$ unless $\nabla f(\theta_t) = 0$, hence allowing us to monotonically decrease f until a local optimum is reached.

Computing λ_t Since (4) is a simple quadratic convex programming, it is easy to see that the solution is $v_t = \nabla f(\theta_t) + \lambda_t \nabla g(\theta_t)$ where λ_t solves the following dual problem of (4):

$$\lambda_t = \arg\min_{\lambda \ge 0} \left\{ \left\|\nabla f(\theta_t) + \lambda g(\theta_t)\right\|^2 - \lambda \phi(\theta_t) \right\} = \max\left(\frac{\phi(\theta_t) - \nabla f(\theta_t)^\top \nabla g(\theta_t)}{\left\|\nabla g(\theta_t)\right\|^2},\quad 0 \right).$$

Practical Choice of ϕ There is a broad range of ϕ's that satisfies (5). Two particularly simple choices stand out which are suitable for constrained and lexico optimization, respectively:

1. $\phi(\theta_t) = \alpha(g(\theta_t) - c)$, where $\alpha > 0$. This puts a strong constraint descent requirement on g when $g(\theta_t) - c$ is large and positive, and allows g to increase increasingly faster when $g(\theta_t) \le c$ and as we move towards the interior of the feasible set. Note that $\nabla g^\top v \ge \alpha(g - c)$ can be viewed as a linearization of the constraint $g(x) \le c$ and hence the algorithm in this case can be viewed as a simple variant of sequential quadratic programming (SQP) which uses an identity matrix to approximate the Hessian of f.

2. $\phi(\theta_t) = \beta \left\|\nabla g(\theta_t)\right\|^2$, where $\beta > 0$. Assume $\nabla g(\theta_t) = 0$ implies global minimum for g, then this ϕ satisfies (5) for lexico case (3), while does not require to estimate the unknown g^*. Because $\phi \ge 0$, it enforces us to strictly decrease g all the time, although with a increasingly lower speed when θ_t approaches the optima of g. Meanwhile, f acts as the secondary objective that is decreased when it does not interfere with the descent on g. In practice, if g has multiple local minima, then the algorithm would minimize $f(\theta)$ subject to $g(\theta) \le g(\hat{\theta}^*)$, where $\hat{\theta}^*$ is a local minimum of g that the algorithm encounters.

Combining the two cases into a single ϕ, we proposes to use

$$\phi(\theta) = \min(\alpha(g(\theta) - \hat{g}),\ \ \beta \left\|\nabla g(\theta)\right\|^2), \tag{6}$$

where we take $\hat{g} = c$ if the goal is to solve the constrainted optimization (1) with threshold c, and take $\hat{g}$ to be any estimate of lower bound of g^* (i.e., $\hat{g} \le g^*$) for solving the lexicographic optimization problem (3); for example, if g is a norm or mean square loss, then $\hat{g} = 0$ is a natural estimation of the lower bound. If no lower bound can be estimated, we can take $\hat{g} = -\infty$, and hence reduce back to $\phi(\theta) = \beta \left\|\nabla g\right\|^2$. Different choices of the hyper-parameters α, β controls the speed of constraint satisfaction vs. loss minimization, which we find has small impact on practical performance on real-world problems . Therefore, in all our experiments, we use $\alpha = \beta = 1$, leaving the algorithm introducing no extra hyper-parameters compared with standard (stochastic) gradient descent. See Algorithm 1 for the algorithm.

Profiling Pareto set The combination of constrained and lexico optimization provides an automatic approach to obtaining uniformly distributed points from the Pareto set of (f, g). To do so, we start with solving the lexico problem (3), which corresponds to one of the end points of the Pareto set; we then walk through the Pareto points by solving the constrained optimization (1) with c increasing linearly from the g^* estimated from lexico optimization. Optionally, we can also solve the mirror version $\min_\theta g(\theta)\ s.t.\ f(\theta) \le f^* := \inf_{\theta'} f(\theta')$ to obtain the other end point, and then solve (1) with c distributed uniformly between the two end points.

Algorithm 1 Dynamic Barrier Gradient Descent for (1) and (3)

Choose stepsize $\{\epsilon_t\}$ and the dynamic barrier function ϕ in (6) (use $\alpha = \beta = 1$ by default; set $\hat{g} = c$ for constrained optimization (1) and $\hat{g}$ to be any value no larger than g^* for lexicographic optimization (3)).

for iteration t **do**

$$\theta_{t+1} \leftarrow \theta_t - \epsilon_t(\nabla f(\theta_t) + \lambda_t \nabla g(\theta_t)), \quad \lambda_t = \max\left(\frac{\phi(\theta_t) - \nabla f(\theta_t)^\top \nabla g(\theta_t)}{\|\nabla g(\theta_t)\|^2}, \quad 0\right).$$

end for

3 Theoretical Analysis

We analyze the convergence of Algorithm 1 for solving both constrained optimization (Section 3.1) and lexico optimization (Section 3.2). A key of our results is that they hold for general non-convex functions, which is essential for deep learning applications. In comparison, we show in Section 3.3 that the commonly used methods that transform (1) into a sequence of unconstrained optimization (2) rely on strong duality assumption and hence unsound for non-convex cases. For simplicity, we focus on the continuous-time limit of the algorithm with $dx_t/dt = -v_t$, where $t \in [0, +\infty)$ denotes the continuous time. The discrete time counterparts hold similarly but require more technical conditions and are deferred to the Appendix.

3.1 Constrained Optimization Case

We analyze the algorithm with a general ϕ satisfying the sign condition (5) for the regular constrained optimization (1) with $c > g^*$. We start with recalling the KKT necessary condition [24]: Assume x^* is a local optimal solution of (1), that f and g are continuously differentiable, and that $\|\nabla g(x^*)\| \neq 0$ (which holds only if $c > g^*$), then there exists a Lagrange multiplier $\lambda^* \geq 0$, such that

$$\textit{Stationarity: } \nabla f(\theta^*) + \lambda \nabla g(\theta^*) = 0, \quad \textit{Feasibility: } g(\theta^*) \leq c, \quad \textit{Slackness: } \lambda(g(\theta^*) - c) = 0.$$

Assume that ϕ satisfies the sign condition (5). For $\lambda \geq 0$, we can use the following score function to assess the satisfaction of the KKT condition:

$$K_\nu(\theta, \lambda) = \|\nabla f(\theta) + \lambda \nabla g(\theta)\|^2 + \nu[\phi(\theta)]_+ + \lambda[-\phi(\theta)]_+, \tag{7}$$

where $\nu > 0$ and $[x]_+ = \max(x, 0)$; it is easy to see that $K_\nu(\theta, \lambda) \geq 0$ for $\theta \in \mathbb{R}^d$ and $\lambda \geq 0$ and $K_\nu(\theta, \lambda) = 0$ iff $(\theta, \lambda) \in \mathbb{R}^d \times [0, \infty)$ meets the KKT condition.

A different way to access the optimality is to use the following L_1 penalty function,

$$P_\mu(\theta) = f(\theta) + \mu[g(\theta) - c]_+, \tag{8}$$

whose minimum points coincide with the solution of (1) if μ is taken to be sufficiently large [24].

The results below show that Algorithm 1 is optimal both in terms of the KKT loss $K_\nu(\theta, \lambda)$ and the penalty function $P_\mu(\theta)$.

Assumption 3.1. *Assume $\|\nabla g(\theta)\| > 0$ on $\bar{G}_c := \{\theta \colon g(\theta) \geq c\}$, which implies that $c > g^*$.*

Assumption 3.2. *Assume f, g are differentiable and $\|\nabla g\|$ is continuous on $\mathbb{R}^d$. Assume $f^* = \inf_{\theta \in \mathbb{R}^d} f(\theta) > -\infty$, and $\sup_{\theta \in \bar{G}_c}(\|\nabla f(\theta)\|, \phi(\theta)) < \infty$.*

Assumption 3.1 ensures that is no local optima of g outside of the interior of the feasible set. This is necessary, since otherwise no local descent algorithm can provably guarantee to reach the feasible set since it may get stuck at the local optima outside the feasible set. Assumption 3.2 are typical regularity conditions.

Theorem 3.3. *Let Assumption 3.1 and 3.2 hold. Consider the continuous-time updates $dx_t/dt = -v_t$ with v_t in* (4) *and any ϕ satisfying the sign condition* (5)*. Then we have $\bar{\lambda} := \sup_t\{\lambda_t \colon g(\theta_t) \geq c\} < +\infty$, and for any $\mu \geq 0$,*

$$\frac{d}{dt} P_\mu(\theta_t) \leq -K_{\mu - \bar{\lambda}}(\theta_t, \lambda_t).$$

Therefore, we have 1) the algorithm monotonically decreases $P_\mu(\theta)$ for $\mu \geq \bar{\lambda}$; 2) the algorithm (θ_t, λ_t) converges to a limit (θ^*, λ^*), which satisfy the KKT condition if ϕ is a continuous function, and 3) $K_\nu(\theta_t, \lambda_t)$ with any $\nu \geq 0$ converges to zero at an $O(1/t)$ rate:

$$\mathbb{E}_{t\sim\mathcal{U}([0,T])}[K_\nu(\theta_t, \lambda_t)] \leq \frac{1}{T}(L_{\nu+\bar{\lambda}}(\theta_0) - f^*),$$

where T is any finite time, and $\mathbb{E}_{t\sim\mathcal{U}([0,T])}[\cdot]$ is the expectation as we draw t uniformly in $[0, T]$.

3.2 Lexicographic Optimization Case

We now analyze our method for lexicographic optimization (3). Note that (3) is a degenerated case of (1), for which the typical KKT condition *does not* apply because $\|\nabla g(\theta)\| > 0$ fails to hold on every point in the feasible set of (3), which is an important constraint quantification condition [24]. Instead, exact optimality conditions of (3) depends on second or even higher order information of g and requires involved technical conditions (see e.g., [8]). We avoid this difficulty and show in the following that a point is lexico optimal if it is a limit of a sequence of increasingly accurate approximate solutions of constrained optimization with $c \to g^*$.

Lemma 3.4. *Let θ_t be the ε_t-optimal solution of $f_t^* := \min_\theta\{f(\theta) \;\; s.t. \;\; g(\theta) \leq c_t\}$ in the sense that $f(\theta_t) \leq f_t^* + \varepsilon_t$, where $c_t \geq g^*$ and $c_t \to g^*$ and $\varepsilon_t \to 0$ as $t \to +\infty$. Assume f is continuous, then all limit points of $\{\theta_t\}$ is a solution of $\min_\theta\{f(\theta) \;\; s.t. \;\; g(\theta) \leq g^*\}$.*

In the following, we show that our method achieve the above in an local optimum sense: our method with a proper ϕ generates a convergent sequence $\{\theta_t\}$ such that 1) θ_t converges to a local optimum θ^* of g, and 2) each θ_t approximately solves the constrained optimization (1) with threshold $c_t := g(\theta_t)$ in the sense of KKT loss:

$$K_0(\theta_t, \lambda_t \mid c_t) := \|\nabla f(\theta_t) + \lambda_t \nabla g(\theta_t)\|^2 \to 0, \quad \text{as } t \to +\infty,$$

where $K_0(\theta, \lambda \mid c_t)$ is the KKT loss (7) of $\min_\theta\{f(x) \;\; s.t. \;\; g(\theta) \leq c_t := g(\theta_t)\}$, where the other remaining two terms in (7) are zero given that $g(\theta_t) - c_t = 0$.

Theorem 3.5. *Let Assumption 3.2 hold with $c = g^*$ and $\sup_{\theta\in\mathbb{R}^d} \|\nabla g\| < \infty$ and $g^* > -\infty$. Following $dx/dt = -v_t$ with v_t in* (4) *with any ϕ that satisfies*

$$0 \leq \phi(\theta_t) \leq \beta \|\nabla g(\theta_t)\|^\tau,$$

where $\beta > 0$ and $\tau \geq 2$, we have for any time $T \geq 0$,

$$\mathbb{E}_{t\sim\mathcal{U}([0,T])}[\phi(\theta_t)] \leq \frac{(g(\theta_0) - g^*)}{T}, \qquad \mathbb{E}_{t\sim\mathcal{U}([0,T])}\left[\|\nabla f(\theta_t) + \lambda_t \nabla g(\theta_t)\|^2\right] \leq \frac{C}{T^{1-1/\tau}},$$

where C is a finite constant depends on β, τ, f, g and θ_0. Furthermore, the sequence $\{\theta_t\}$ converges to a limit θ^*; and θ^* satisfies $\|\nabla g(\theta^*)\| = 0$ if ϕ satisfies the sign condition (5) *and is continuous.*

We can see that the power index τ controls the relative convergence speed of $\phi(\theta_t)$ (measuring the minimization of g), and the KKT loss (measuring the minimization of f). If $\phi(\theta) = \beta \|\nabla g(\theta)\|^\tau$ with $\tau \geq 2$, we have $\mathbb{E}_{t\in\mathcal{U}([0,T])}[\|\nabla g(\theta_t)\|^2] = O(1/T^{2/\tau})$ and $\mathbb{E}_{t\in\mathcal{U}([0,T])}[\|v_t\|^2] = O(1/T^{1-1/\tau})$. The rate is slower than the regular constrained optimization due to the degeneration caused by $\|\nabla g(\theta)\| = 0$ on the feasible set.

3.3 Non-convexity and Primal vs. Dual Methods

Many popular approaches for constrained optimization (1) used in machine learning are based on transforming it into a sequence of unconstrained optimization of form (2) with λ iteratively updated in certain way by viewing it as the Lagrange multiplier. However, all these methods based on (2) are fundamentally *dual methods* which are guaranteed to work only when f, g are convex and the strong duality holds. In comparison, Algorithm 1 is a *primal method*, which directly solves the primal problem and is hence more suitable for non-convex functions.

To elaborate, note that the constrained optimization is equivalent to the following minimax problem:

$$\min_{\theta\in\mathbb{R}^d} \max_{\lambda\geq 0} f(\theta) + \lambda g(\theta). \tag{9}$$

If the strongly duality holds, exchanging the order of min and max yields a equivalent dual problem:

$$\max_{\lambda \geq 0} \Phi(\lambda), \qquad \Phi(\lambda) = \min_{\theta \in \mathbb{R}^d} \{f(\theta) + \lambda g(\theta)\}. \tag{10}$$

Therefore, one can estimate λ^* by maximizing $\Phi(\lambda)$ with gradient ascent, yielding the *dual ascent* algorithm. Because calculating $\Phi(\lambda)$ requires to solve the whole unconstrained optimization (2) and is costly, a simplification is to alternate between the gradient ascent on λ and gradient descent on θ for solving the minimax problem, yielding the following *primal-dual gradient method*:

$$\lambda_t \leftarrow [\lambda_{t-1} + \xi g(\theta_t)]_+, \qquad \theta_{t+1} \leftarrow \theta_t - \epsilon(\nabla f(\theta_t) + \lambda_t \nabla g(\theta_t)), \tag{11}$$

where ϵ and ξ are step sizes.

Unfortunately, all these methods, which *decouple* the estimation of θ and λ, are not suitable for non-convex functions due to the lack of strong duality. In fact, when f and/or g are sufficiently non-convex, *the true solution θ^* can be a local maximum (rather than local minimum), or saddle point of $f(\theta) + \lambda^* g(\theta)$*, which happens once $\nabla^2 f(\theta^*) + \lambda^* \nabla^2 g(\theta^*)$ is not positive (semi)-definite, where λ^* is the Lagrangian multiplier associated with θ^* in the KKT condition. In this case, even when we know the true Lagrange multiplier λ^*, maximizing $f(\theta) + \lambda^* g(\theta)$ as advocated in (2) would not yield the true solution θ^*. Similarly, the primal-dual update in (11) is doomed to fail because one can show that (θ^*, λ^*) is guaranteed to be an unstable fixed point of (11) (see Appendix). The augmented Lagrange methods [3], which amount to replacing the objective f with $f_\rho(\theta) := f(\theta) + \rho[g(\theta) - c]_+^2$, $\rho > 0$, can help "convexify" the objective if $[g(\theta) - c]_+^2$ is convex (such as the case of linear constraints), but may not help when g and $[g - c]_+^2$ are non-convex. In comparison, Algorithm 1 works for general non-convex functions we show in Section 3.1-3.2.

Example 3.6. *Consider a toy problem:* $\min_{\theta \in \mathbb{R}} |\theta - 1|^\alpha \;\; s.t. \;\; |\theta|^\alpha \leq b^\alpha$ *with* $b \in [0, 1)$ *and* $\alpha > 0$. *Note that all* $\alpha > 0$ *yields an equivalent problem whose true solution is* $\theta^* = 1 - b$ *and the associate Lagrange multiplier* $\lambda^* = (1 - b)^{\alpha - 1}/b^{\alpha - 1}$. *However, the choice of* α *changes the convexity of the problem. One can show that if* $\alpha < 1$*, the true solution is a local maximum of* $f(\theta) + \lambda^* g(\theta)$*, and hence methods based on minimizing the linear combination* (2) *fail (in fact, they fail even if* $\alpha = 1$*). However, Algorithm 1 solves the problem correctly for all* $\alpha > 0$. *See Appendix for more analysis.*

3.4 Related Works

Constrained Optimization There is a large classical literature on constrained optimization [24, 3, 4], including the primal methods such as the penalty function methods, interior point methods, sequential quadratic programming (SQP), and Lagrange multiplier and dual methods. Our method is mostly related to SQP [24], of which Algorithm 1 can be viewed as a special case if we take $\phi = \alpha(g - c)$. However, our method and analysis allow a broader class of ϕ with a general dynamic barrier view, especially with the gradient based ϕ for the lexico case; importantly, Algorithm 1 is fast and very simple to implement, making it well suitable for large scale deep learning tasks, which is our main target domain. In comparison, the traditional SQP are often complicated/expansive and have found no wide applications in large-scale deep learning tasks. The penalty function methods are widely used, but require to use a large penalty coefficient which may cause numerical issues. There is a vast literature on Lagrange multipler and (primal-)dual methods [3], which, however, are designed for convex problems and problematic for non-convex problems as we show in Section 3.3.

Lexicographic Optimization Despite offering a promising tool for incorporating secondary loss functions, lexico optimization (3) has been rarely known and studied in machine learning. In the bilevel optimization literature, (3) is known as simple bilevel programming (SBP), which is a special case of optimization on efficient sets, and more generally mathematical program with equilibrium constraints (MPEC) [9, 8]. A number of algorithms (e.g., [26, 30, 29]) have been proposed in the optimization literature, which, however, focus on convex functions and yield double loop algorithms that repeatedly call an inner loop optimization, which are more expensive than our method.

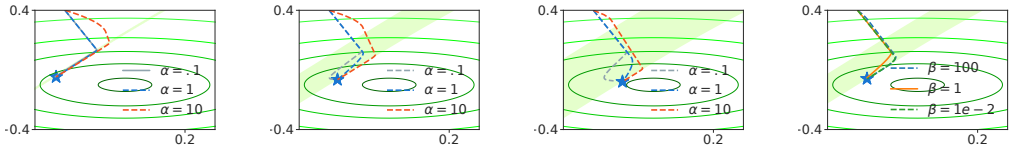
Approximating Pareto Set The method of constructing Pareto sets using constrained optimization is known as the ε-constrained method in the multi-objective optimization literature (e.g., [21]), which improves over the linear scalarization approach (2) for non-convex Pareto sets. However, traditional ε-constrained method does not consider the lexico case and hence need extra steps/information to decide the range of the threshold c. There are a vast literature of other approaches that work for non-convex Pareto sets in the MOO literature [22, 14, 25], which has different advantages and drawbacks w.r.t. the ε-constrained method. The goal of this work is to improve constrained and lexico optimization, rather than investigating different MOO methods.

4 Experiments

Our simple method can find broad applications in various deep learning tasks. We show that 1) the constrained+lexico optimization approach provides an efficient approach to profile non-convex Pareto sets, as we demonstrate in a fairness classification and a sparse representation learning task; and 2) lexico optimization alone offers a parameter-free and stable approach to incorporate auxiliary regularization/loss without hurting the optimization of the main loss function, as we show in a variety of tasks: fine-grained image classification, semi-supervised learning and semantic segmentation.

Hyperparameters and Settings We use Algorithm 1 with ϕ in (6). We always set $\alpha = \beta = 1$ except ablation studies and hence introduces no extra hyper-parameters beyond regular (stochastic) gradient descent. For step size, we fix the learning rate to be 10^{-2} when profiling Pareto sets, and use the default step size scheduler of the baseline methods of each of the other applications. All the results and error bars are estimated over 5 random trails.

2D Toy We use an $\mathbb{R}^d$-toy example in Figure 1, with $f(\theta) = \|\theta - \theta^*\|^2$ and $g(\theta) = \left\|a^\top \theta + b\right\|^2$, where $\theta = [\theta_1, \theta_2]$ and $\theta^*, a \in \mathbb{R}^2$, $b \in \mathbb{R}$ are constants. Note that the optima set of g is the line $a^\top \theta + b = 0$. Figure 1 shows that our method with different (α, β) all converge to the true solution for different threshold c, with larger values of (α, β) yielding faster descent on the constraint.



(a) $c = g^* = 0, \beta = 1$ (b) $c = 0.01, \beta = 1$ (c) $c = 0.1, \beta = 1$ (d) $c = g^* = 0, \alpha = 5$

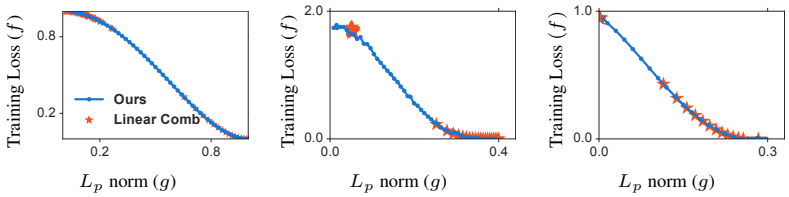
Figure 1: Algorithm trajectories on 2D toy with different threshold c and (α, β) in (6). The contours denote the objective f and the green areas are the feasible set. The stars are true solutions.

Sparsity Representation learning We learning sparse feature representations on a supervised dataset $\mathcal{D}$ of (x, y) pairs by applying a non-convex L_p regularization:

$$f(\theta) = \mathbb{E}_{\mathcal{D}}[\ell(y - \phi_\theta(h_\theta(x))^2)], \qquad g(\theta) = \mathbb{E}_{\mathcal{D}}[\|h_\theta(x)\|_p^p],$$

where $\ell(\cdot, \cdot)$ is the data loss, $h_\theta(x) \mapsto z \in \mathbb{R}^m$ is a trainable feature map, ϕ_θ is a prediction head, and p is a power coefficient. We choose $p = 0.5$ which yields a non-convex sparsity penalty. We choose h_θ: to be a simple linear model on a simulated dataset (Figure 2(a)) and the pretrained BERT-base model [10] on the IMDb binary classification dataset (Figure 2(b)) and on the 20 news group (NG20) dataset (Figure 2 (c)). NG20 is processed into 4-class classification following [12]. For IMDb, we randomly sample 2K examples from the full training dataset.

We profile the Pareto set by first finding the two end points with lexico optimization (3) and its mirror version and then solving the constrained optimization (1) with c distributed evenly between the two end points (see Section 2). Figure 2 shows that our method (blue points) profiles the Pareto fronts evenly, which are all non-convex. The red stars are obtained by gradient descent on the linear combination (2), with $\omega := \lambda/(1 + \lambda)$ taken from the uniform grid of $[0, 1]$, which can not profile the whole Pareto sets and yield unevenly distributed points as theory predicts.



(a) Linear on Toy Data (b) BERT-base on NG20 (c) BERT-base on IMDb

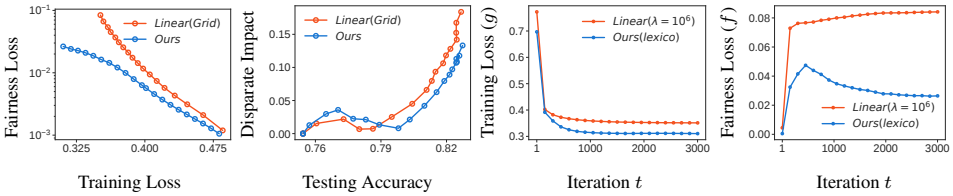
Figure 2: Our method (blue dots) profiles the Pareto fronts evenly in sparse representation learning, while linear combination (red stars) only finds parts of the Pareto front and yields unevenly distributed points.

Fairness Regularization Standard ML models may yield unfair prediction in real world systems, making it necessary to regularize the training with fairness metrics. Assume we want to learn a predictor $\hat{y}_\theta(x)$ on a dataset $\mathcal{D}$ of (x, y, a) pairs, where x, y are feature and label, and a a protected demographic attribute (e.g., male vs. female). We consider fairness-regularized learning with

$$f(\theta) = (\mathrm{cov}_{\mathcal{D}}(\hat{y}_\theta(x), a))^2, \qquad g(\theta) = \mathbb{E}_{\mathcal{D}}[\ell(y, \hat{y}_\theta(x))],$$

where f measures fairenss using the covariance between the protected attribute a and the prediction $\hat{y}_\theta(x)$; it can be viewed as a continuous surrogate of the *disparate impact* criterion [5]. g is the standard classification loss. For the experiments, we use the *Adult Income* dataset [15] for predicting whether the annual income of a person is $\geq \$50,000$, with gender as the predicted attribute. Following the setting in [20, 17], we randomly sample a subset of 5,000 data points from the original training set as our training set.

Figure 3 (a,left) shows the Pareto front profiled by our method (blue dots) and by gradient descent on the linear combination loss (2) (red dots) with $\lambda/(1+\lambda)$ uniformly taken from a 20×1 grid of $[0, 1]$. We can see that our method finds a good non-convex Pareto set, while (interestingly) the linear combination method finds a different convex, but sub-optimal Pareto set, in a different local mode of the landscape. Note that since the our Pareto front is concave, the points on it are the local maxima (not minima) of the linear combination loss (2), and hence can not be found by gradient descent on (2) even if the corresponding λ is given. Figure 3(a,right) shows the prediction accuracy on the true disparate impact score on the testing data; Figure 3(b) shows the trajectory of f and g in Algorithm 1 for the Lexico case (3), and the gradient descent of (2) with a very large λ



(a) Pareto Front on Training & Testing Sets (b) Training curves of f and g.

Figure 3: Fairness regularization on Adult Dataset.

Lexicographic ℓ_2 Regularization Lexico optimization (3) provides a parameter-free approach to adding secondary regularization without hurting the optimization of the main loss. We apply it to fine-tune ImageNet pretrained models on fine-grained image classification with L_2 regularization:

$$f(\theta) = \|\theta\|_2^2, \qquad g(\theta) = \mathbb{E}_{\mathcal{D}}[\ell(\hat{y}_\theta(x), y)],$$

where the main loss g is the typical training loss, and the secondary loss f is the squared L_2 norm. Recent works [e.g. 13, 27] have shown that the L_2 regularization (when combined linearly as (2)) helps improve the accuracy with λ optimally selected with a holdout data (which causes high training cost). Lexico optimization eliminates the need of tuning λ.

For our experiments, we use two fine-grained classification datasets: Oxford Flower [23] and Stanford Car [16], with two deep neural architectures, including EfficientNet [31], and AlphaNet [32]; AlphaNet is the state-of-the-art mobile model on ImageNet. We use the official checkpoints provided by [31] and [32]. Table 1 shows that the Lexico approach, which requires no tuning of λ, is almost in par with the best results of the linear combination approach with the best λ on a grid, which is given by the baseline code of [13, 27].

		ℓ_2 regularization (linear combination)					Lexico
		∞	2.5×10^7	2.5×10^6	2.5×10^5	2.5×10^4	
Oxford Flower	AlpahNet-A0	96.6±0.2	96.8±0.2	97.2±0.1	**97.7±0.2**	97.3±0.1	**97.6±0.1**
	EffNet-B0	96.4±0.1	96.7±0.1	96.9±0.1	**97.2±0.1**	96.9±0.1	**97.1±0.1**
Stanford Car	AlpahNet-A0	**91.6±0.1**	91.3±0.2	91.0±0.1	90.8±0.2	90.6±0.2	**91.6±0.1**
	EffNet-B0	**91.0±0.1**	90.7±0.2	90.6±0.2	90.6±0.1	90.3±0.1	**91.0±0.1**

Table 1: Top-1 test accuracy on fine-grained image classification tasks obtained by lexicographic optimization, and the linear combination with different λ.

Lexico Optimization for Semi-Supervised Learning Recently, various unsupervised auxiliary losses based on data augmentation have been used in semi-surprised learning [1, 2, 11, 28, 34, 35]. Typically, the auxiliary loss is linearly combined with the main supervised loss. Instead this approach, we combine these two using lexico optimization. Following [e.g. 34], we consider

$$f(\theta) = \mathbb{E}_{\mathrm{T}\sim\mathcal{T}}\mathbb{E}_{x\sim\mathcal{D}_u}[\mathrm{KL}(\hat{y}_\theta(x) \,||\, \hat{y}_\theta(\mathrm{T}(x)))], \qquad g(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_s}[\ell(\hat{y}_\theta(x), y)],$$

where the main loss g is the typical training loss on a labeled dataset $\mathcal{D}_s$ of (x, y) pairs, and the secondary loss f is the unsupervised data augmentation (UDA) loss [e.g. 34] on an unlabeled dataset $\mathcal{D}_u$ consisting of x only; intuitively, f measures the consistency of the prediction from x and a perturbed $x' = \mathrm{T}(x)$ using a randomly transform T (e.g., rotation, flipping, etc). Table 2 shows that the result of lexico optimization is comparable with or even better than the baseline provided by [e.g. 34] which uses linear combination with λ gradually decayed during training from a very large value (e.g. ∞) to 0.2. The improvement of the lexico method is specially significant when the size of labeled dataset is small (e.g. 100).

Method	CIFAR-10			CIFAR-100		
	100 labels	250 labels	1000 labels	1000 labels	2500 labels	10000 labels
UDA	75.28±2.39	91.18±1.08	**94.56±0.21**	54.85±0.83	66.37±0.28	**67.42±0.22**
UDA + Lexico	**77.61±1.83**	**91.86±0.85**	**94.53±0.15**	**56.91±0.78**	**67.06±0.26**	**67.57±0.14**

Table 2: Testing accuracy (%) of different methods on CIFAR-100 and CIFAR-10.

Lexico Optimization for Semantic Segmentation Recent state-of-the-art semantic segmentation methods are typically trained with the linear combination of loss functions from two models with different complexity/resolutions [6]:

$$f(\theta) = \mathbb{E}_{\mathcal{D}}[\ell(\tilde{y}_\theta(x), y)], \qquad g(\theta) = \mathbb{E}_{\mathcal{D}}[\ell(\hat{y}_\theta(x), y)],$$

where $\hat{y}_\theta$ is the main prediction model and $\tilde{y}_\theta$ an auxiliary model with lower complexity and cost. Table 3 shows that lexico optimization yields better or comparable results on these two losses than linear combination. Here we use Swin Transformer [18], a recent-proposed state-of-the-art model, as our backbone and use the code provided by [6]. We use the Cityscapes [7] benchmark, and calculate the main loss with UperNet [33], and use FCN [19] on the 3/4 of the total depth of the network to calculate the auxiliary loss. The grid space of linear combination is the default one provided by [6].

Model	Linear Combination λ					Lexico
	10	5	10/3	2.5	2	
Swin-Base UperNets	79.3±0.3	79.6±0.2	79.5±0.1	79.5±0.1	79.1±0.3	**79.7±0.1**
Swin-Base FCN	78.3±0.2	78.2±0.1	78.1±0.2	78.2±0.2	78.0±0.2	**78.5±0.1**

Table 3: mIoU for segmentation on Cityscapes validation set. All the models are trained with 40K iterations and batch size 8. The results are averaged over five trials.

5 Conclusion, Limits, Future Directions

We provide a joint treatment of constrained and lexicographic optimization with a general dynamic barrier gradient descent algorithm, which provides an efficient tool for profiling non-convex Pareto sets and incorporating secondary losses as we show in experiments. Our work has potential positive social impact for providing new tools for more efficient and fairness ML, but we will have no control of malicious use of our open source code after it is released.

Currently, our work focuses on a single constraint, which yields a particularly algorithm, and is sufficient for broad applications. The extension to multiple constraints is straightforward, but the calculation of λ_t (which would be a vector) requires to solve a convex quadratic programming, which can be expensive for deep learning tasks when the number of constraints are large. Future works will develop computationally efficient methods for handling a large number of constraints.

For theoretical analysis, our results assume an extract gradient, while practical applications use stochastic gradient descent (SGD). A future direction is to extend the analysis to the SGD setting.

References

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

[2] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

[3] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[4] Joseph-Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia A Sagastizábal. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[5] Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010.

[6] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[8] S Dempe, N Dinh, and J Dutta. Optimality conditions for a simple convex bilevel programming problem. In *Variational analysis and generalized differentiation in optimization and control*, pages 149–161. Springer, 2010.

[9] Stephan Dempe, Nguyen Dinh, Joydeep Dutta, and Tanushree Pandit. Simple bilevel programming and extensions. *Mathematical Programming*, pages 1–27, 2020.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Chengyue Gong, Dilin Wang, and Qiang Liu. Alphamatch: Improving consistency for semi-supervised learning with alpha-divergence. *arXiv preprint arXiv:2011.11779*, 2020.

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *The International Conference on Learning Representations*, 2016.

[13] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018.

[14] C-L Hwang and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media, 2012.

[15] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

[16] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[17] Suyun Liu and Luis Nunes Vicente. Accuracy and fairness trade-offs in machine learning: A stochastic multi-objective approach. *arXiv preprint arXiv:2008.01132*, 2020.

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[20] Natalia Martinez, Martin Bertran, and Guillermo Sapiro. Minimax pareto fairness: A multi objective perspective. In *International Conference on Machine Learning*, pages 6755–6764. PMLR, 2020.

[21] George Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, 213(2):455–465, 2009.

[22] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[23] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[24] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[25] Panos M Pardalos, Antanas Žilinskas, and Julius Žilinskas. *Non-convex multi-objective optimization*. Springer, 2017.

[26] Shoham Sabach and Shimrit Shtern. A first order method for solving convex bilevel optimization problems. *SIAM Journal on Optimization*, 27(2):640–660, 2017.

[27] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *arXiv preprint arXiv:2007.08489*, 2020.

[28] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[29] Mikhail Solodov. An explicit descent method for bilevel convex optimization. *Journal of Convex Analysis*, 14(2):227, 2007.

[30] Mikhail V Solodov. A bundle method for a class of bilevel nonsmooth convex minimization problems. *SIAM Journal on Optimization*, 18(1):242–259, 2007.

[31] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[32] Dilin Wang, Chengyue Gong, Meng Li, Qiang Liu, and Vikas Chandra. Alphanet: Improved training of supernet with alpha-divergence. *arXiv preprint arXiv:2102.07954*, 2021.

[33] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018.

[34] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[35] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]